Invariants for equivalence of finite automata

Mike Spivey

Let $S = (S, \Sigma, \delta_S, s_0, F_S)$ and $\mathcal{T} = (T, \Sigma, \delta_T, t_0, F_T)$ be two deterministic finite automata with the same alphabet Σ. We say S and $\mathcal{T}$ are *equivalent* if, for each word $w \in \Sigma^*$, we have $\delta_S(s_0, w) \in F_S$ if and only if $\delta_T(t_0, w) \in F_T$. In other words, S and $\mathcal{T}$ are equivalent if $R \subseteq E$, where

$$R = \{ (\delta_S(s_0, w), \delta_T(t_0, w)) \mid w \in \Sigma^* \},$$

is the set of pairs of states (s, t) that are reached from (s_0, t_0) via the same word w, and

$$E = \{ (s, t) \mid s \in F_S \Leftrightarrow t \in F_T \}.$$

is the set of pairs (s, t) such that s is a final state of S if and only if t is a final state of T. In another characterisation, R is the smallest set that contains (s_0, t_0) and is closed under Δ, where

$$\Delta(s, t) = \{ (\delta_S(s, a), \delta_T(t, a)) \mid a \in \Sigma \}.$$

Thus, to prove $R \subseteq V$ for some set V, it suffices to show that $(s_0, t_0) \in V$ and V is closed under Δ.

Now let us assume that S and T are disjoint and let $X = S \cup T$ and $\delta = \delta_S \cup \delta_T$ and $F = F_S \cup F_T$. Then the problem of determining whether S and $\mathcal{T}$ are equivalent amounts to determining whether the states s_0 and t_0 are equivalent in a certain sense in this joined automaton. We can extend the definition of Δ to $X \times X$ by writing

$$\Delta(x, y) = \{ (\delta(x, a), \delta(y, a)) \mid a \in \Sigma \},$$

and R remains the smallest relation containing (s_0, t_0) that is closed under this operation.

Now observe that we can extend the relation $E \subseteq S \times T$ to an equivalence relation E' on X, with $(x, y) \in E'$ if and only if $x \in F \Leftrightarrow y \in F$. Immediately, $E = E' \cap (S \times T)$, so for $s \in S$ and $t \in T$, we have $(s, t) \in E$ if and only if $(s, t) \in E'$. Consequently $R \subseteq E$ if and only if $R \subseteq E'$.

We'll use the notation $\overline{Q}$ for the *equivalence closure* of a relation Q, that is, the smallest equivalence relation that contains Q. If we take Q as the set of pairs that have been submitted to the *Union* operation in a disjoint-sets data structure, then $\overline{Q}$ is the equivalence relation that is tested with *Find*(x) = *Find*(y). This closure operation has the property that if U is an equivalence relation, and Q is any relation, then $Q \subseteq U$ if and only if $\overline{Q} \subseteq U$.

```
Q := ∅; W := {(s0, t0)};
while W ≠ ∅ do
  extract (u, v) from W;

  if (u, v) ∉ E then
    return FALSE
  end;

  if (u, v) ∉ Q̄ then
    Q := Q ∪ {(u, v)};
    W := W ∪ Δ(u, v)
  end
end;
return TRUE
```

Figure 1: *Program for testing equivalence*

In the algorithm we are about to study, the aim is to find a relation $Q \subseteq E$ such that $(s_0, t_0) \in Q$ and $\overline{Q}$ is closed under Δ. If we succeed, then it follows from minimality of R that $R \subseteq \overline{Q} \subseteq E'$, and so $R \subseteq E$ and the automata are equivalent. If we fail, it will be by finding a pair $(s, t) \in R$ such that $(s, t) \notin E$, and it follows that the automata are not equivalent.

Now consider the algorithm shown in Figure 1, which operates on two binary relations on states, Q and W, with W functioning as a 'work-list' of pairs found in R that are waiting to be examined. I claim that the following are invariants:

(i) $(s_0, t_0) \in \overline{Q} \cup W$,

(ii) $W \subseteq R$,

(iii) $Q \subseteq E$,

(iv) If $(s, t) \in Q$ then $\Delta(s, t) \subseteq \overline{Q} \cup W$.

For most of these relationships, it is easy to check that they are maintained. For invariance of (i), observe that initially $(s_0, t_0) \in W$, and the value of $\overline{Q} \cup W$ only increases: if (u, v) is chosen and removed from W, then the loop body ensures that $(u, v) \in Q$.

For (ii), observe that in each iteration, (u, v) is chosen from W and we may assume $W \subseteq R$, so the new value of W, namely $W' = W \setminus \{(u, v)\} \cup \Delta(u, v)$, is also a subset of R.

For (iii), simply observe that we test $(u, v) \in E$ before adding the pair to Q.

As for (iv), since $\overline{Q} \cup W$ only increases, we need worry only about a new pair (u, v) added to Q; but the adjustment to W deals with keeping this invariant true.

If the loop body returns *FALSE*, invariant (ii) tells us that (u, v) is in R but not E, so S and $\mathcal{T}$ are not equivalent.

If the loop terminates successfully, we have $W = \emptyset$, and invariant (iv) tells us that

$$\text{If } (s, t) \in Q \text{ then } \Delta(s, t) \subseteq \overline{Q}.$$

We can write this property as $Q \subseteq U$, where

$$U = \{\, (x, y) \mid \Delta(x, y) \subseteq \overline{Q} \,\}.$$

Now U is an equivalence relation, so it follows by the properties of equivalence closure that $\overline{Q} \subseteq U$ also, and so $\overline{Q}$ is closed under Δ. But also $(s_0, t_0) \in \overline{Q}$, so $\overline{Q}$ includes all of R, and the argument given above shows that S and $\mathcal{T}$ are equivalent.

Observing that Q is not used in the program other than via the operations

$Q := \emptyset$,

$Q := Q \cup \{(s, t)\}$,

$(s, t) \in \overline{Q}$,

we see that Q can be implemented with a disjoint sets data structure.